Data Analysis and Visualization in R (IN2339)

Exercise Session 5 - High dimensional visualization

Daniela Klaproth-Andrade, Felix Brechtmann, Julien Gagneur

Section 00 - Getting ready

1. Make sure you have already installed and loaded the following libraries:

```
library(ggplot2)
library(data.table)
library(magrittr)   # Needed for %>% operator
library(tidyr)
library(GGally)

library(pheatmap)
library(mclust)
```

Section 01 - Visualizing multiple variables

Gene expression measures the abundance of RNAs per gene. It is indicative of how active a gene is in a sample. Variations of gene expression across samples are indicative of the gene's role. The gene expression data in `cancer_data.rds` is a matrix of gene expression values of 20 genes across 30 tumor samples aimed at understanding the potential role of genes in cancer.

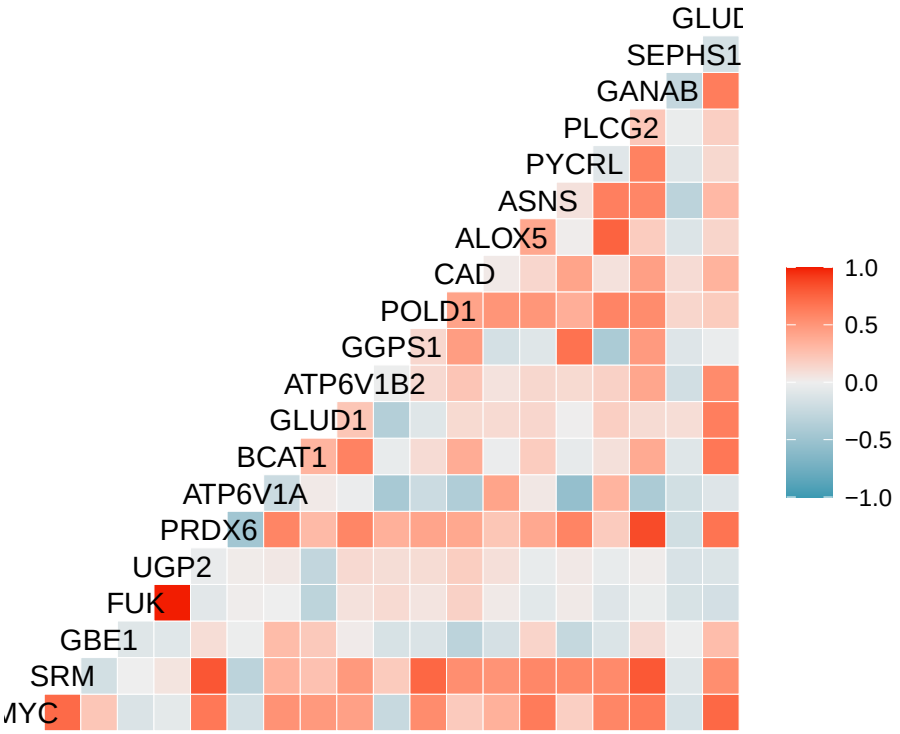
Load the gene expression data in `cancer_data.rds` as a `data.table` with the following line of code:

```
expr <- readRDS("extdata/cancer_data.rds") %>% as.data.table(keep.rownames="tumor_type")
head(expr[, 1:6])
```

```
##     tumor_type        MYC         SRM       GBE1         FUK        UGP2
## 1:       DOHH2 -0.4950154  0.20907327 -0.4366726 100.0000000 100.0000000
## 2:      FARAGE -0.4913630 -0.35590874  0.5322076   0.1037538  -0.6494896
## 3:          HT -0.4559935 -0.23852493  0.6714241  -1.0136358  -0.9651291
## 4:    Kapas231  1.6152666  0.92666297  1.0168147  -0.9978958   1.0064845
## 5:     OCI-LY1  0.4942733  2.40062582 -1.4255166   1.0676067   0.9862178
## 6: OCI-LY1-B50  0.1940777 -0.04941387 -0.4810955   0.2423189   0.7046970
```

1. We are interested in the correlations between genes. Plot the pairwise correlations of the variables in the dataset. Which pair of genes has the highest correlation? *Hint:* remember that you can exclude a column `"colA"` from a data table `DT` with `DT[, -"colA"]`.

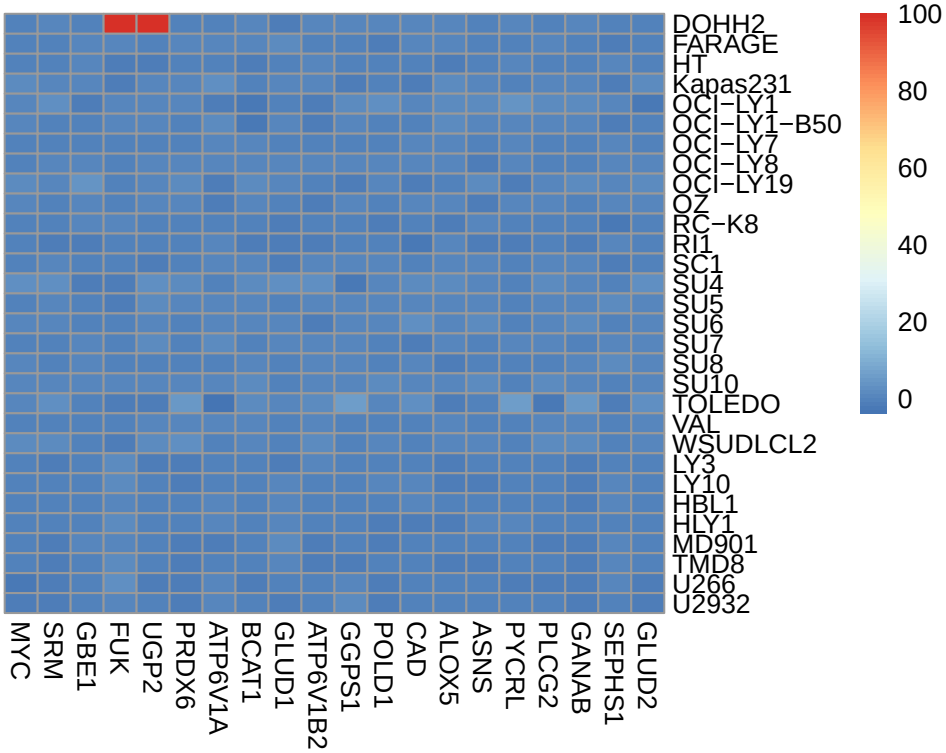
```
# Plot pairwise correlations
ggcorr(expr[, !"tumor_type"])
```




```
# We see a strong correlation between FUK and UGP2
```

2. Visualize the raw data in a heatmap with `pheatmap`.

```
expr_mat <- as.matrix(expr[, !"tumor_type"])
rownames(expr_mat) <- expr[, tumor_type]
pheatmap(expr_mat, cluster_rows = F, cluster_cols = F)
```

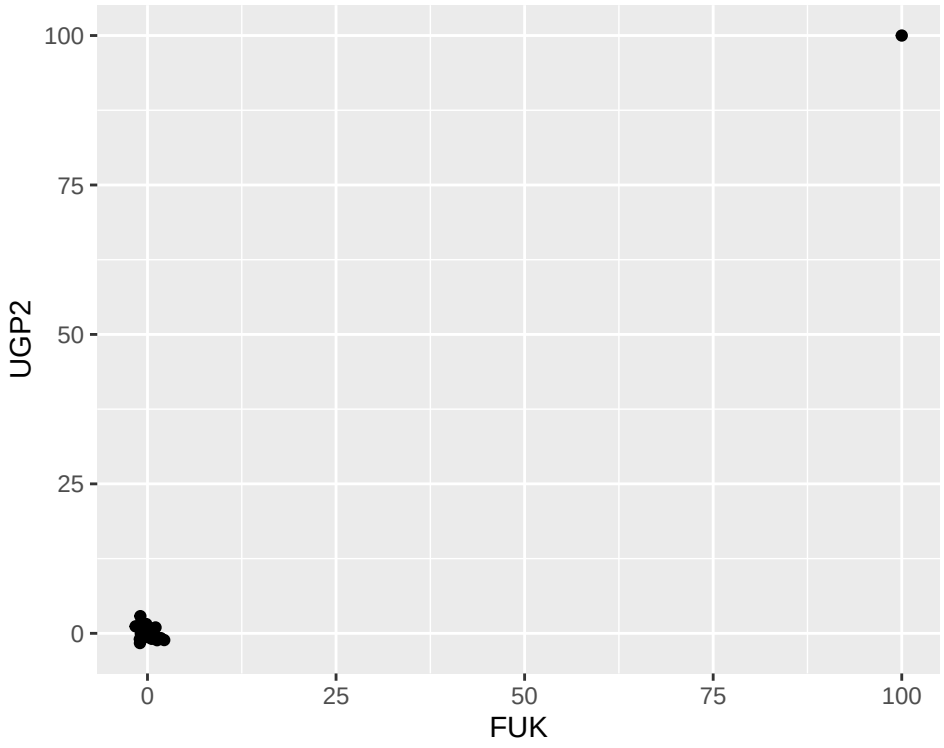


3. Does the latter plot suggest some erroneous entries? Could they have affected the correlations? Check by using an appropriate plot the impact of these erroneous entries on the correlations. Substitute the erroneous values with missing values (NA) and redo the previous questions 1 and 2.

```
# obviously two data points are completely off -->  find them
expr_melt <- melt(expr, id.vars='tumor_type')
expr_melt[order(-value)]
```

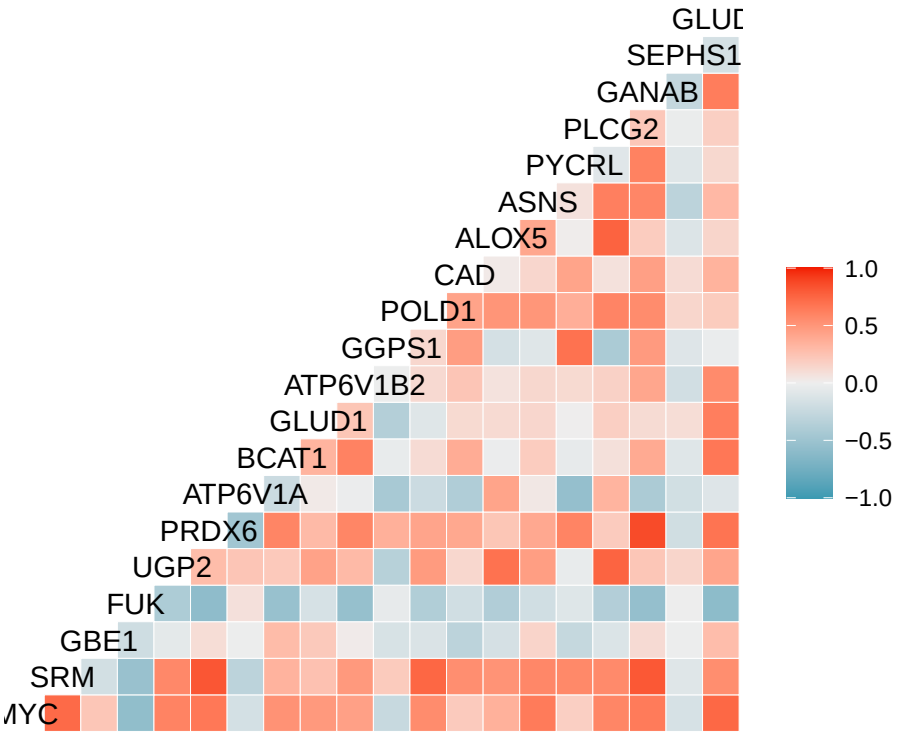
```
##      tumor_type variable      value
##   1:      DOHH2      FUK 100.000000
##   2:      DOHH2     UGP2 100.000000
##   3:     TOLEDO    GGPS1   5.876502
##   4:     TOLEDO    PYCRL   5.555931
##   5:     TOLEDO    PRDX6   4.789574
##  ---
## 596:       U266      MYC  -2.382148
## 597:      RC-K8   SEPHS1  -2.576766
## 598:        SU4    GGPS1  -2.592600
## 599:     TOLEDO    PLCG2  -2.793202
## 600:     TOLEDO  ATP6V1A  -4.058642
```

```
# the expression value of the tumor type  DOHH2 for the genes FUK and UGP2 seems
# to be way too high.

# These look like outliers. Let's plot them to check
# we would like to plot the expression value of genes FUK and UGP2
# It's better to use the original data table for this
ggplot(expr, aes(FUK, UGP2)) + geom_point()
```

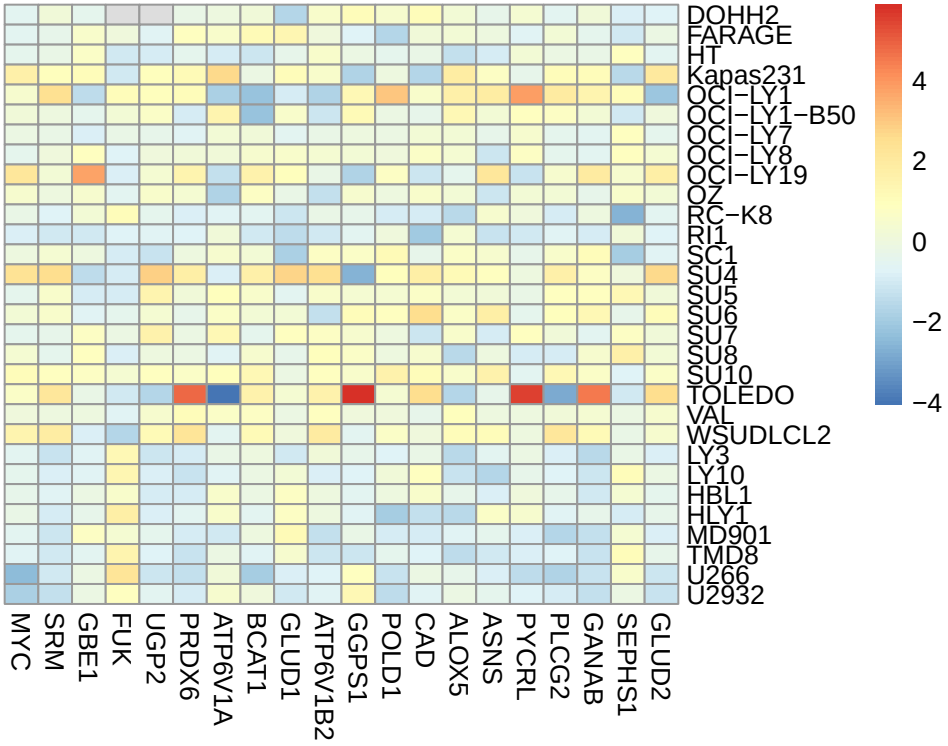



```
# We can substitute the outlier with NA
expr[tumor_type == "DOHH2", FUK := NA]
expr[tumor_type == "DOHH2", UGP2 := NA]
```

```
# Now check the correlation again
ggcorr(expr[, !"tumor_type"])
```



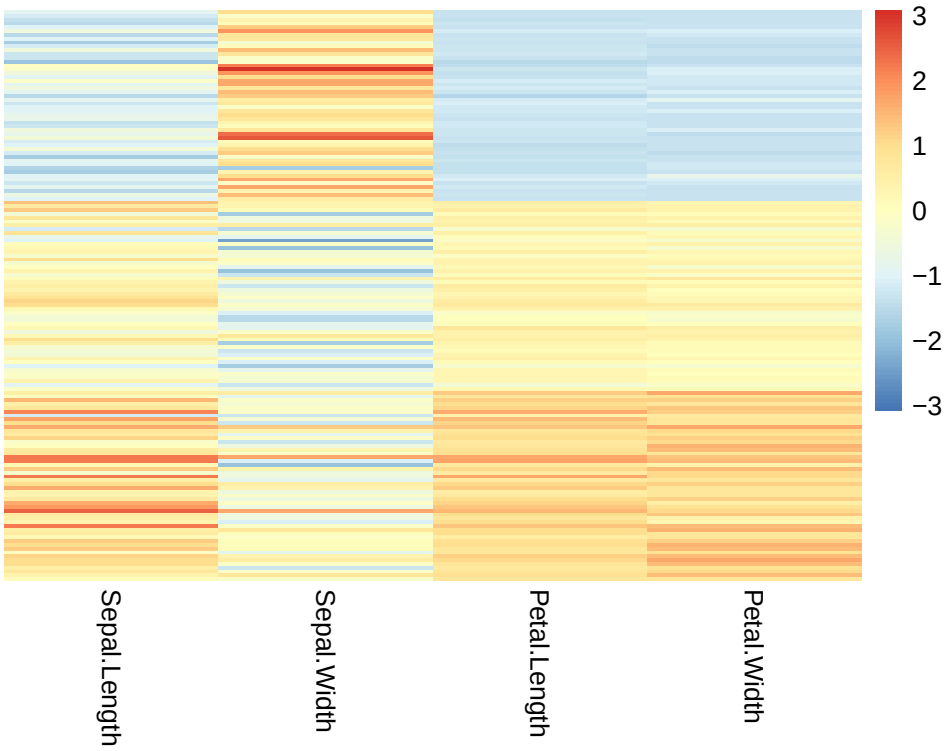

```
# Now plot heatmap again
expr_mat <- as.matrix(expr[, !"tumor_type"])
rownames(expr_mat) <- expr[, tumor_type]
pheatmap(expr_mat, cluster_rows = F, cluster_cols = F)
```



Section 02 - Heatmaps and Hierarchical clustering

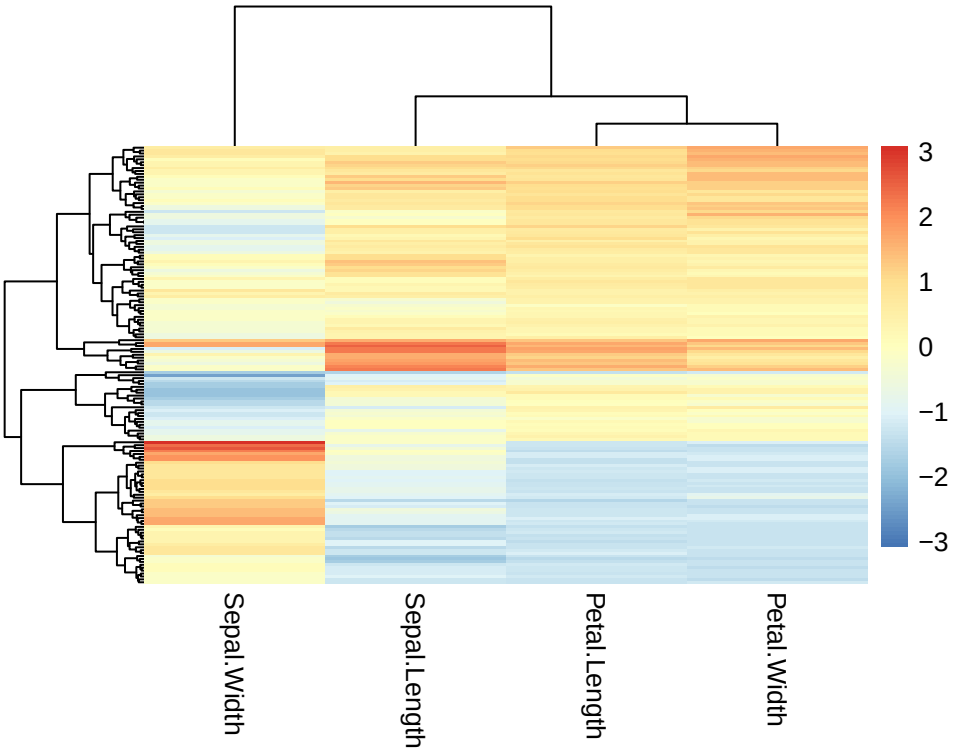
1. Consider the full `iris` data set without the `Species` column for clustering. Create a pretty heatmap with the library `pheatmap` of the data without clustering.

```
# Exclude Species column
plot.data <- iris[, -5]
pheatmap(plot.data, show_rownames=F,
         cluster_rows=F, cluster_cols=F, scale='column')
```



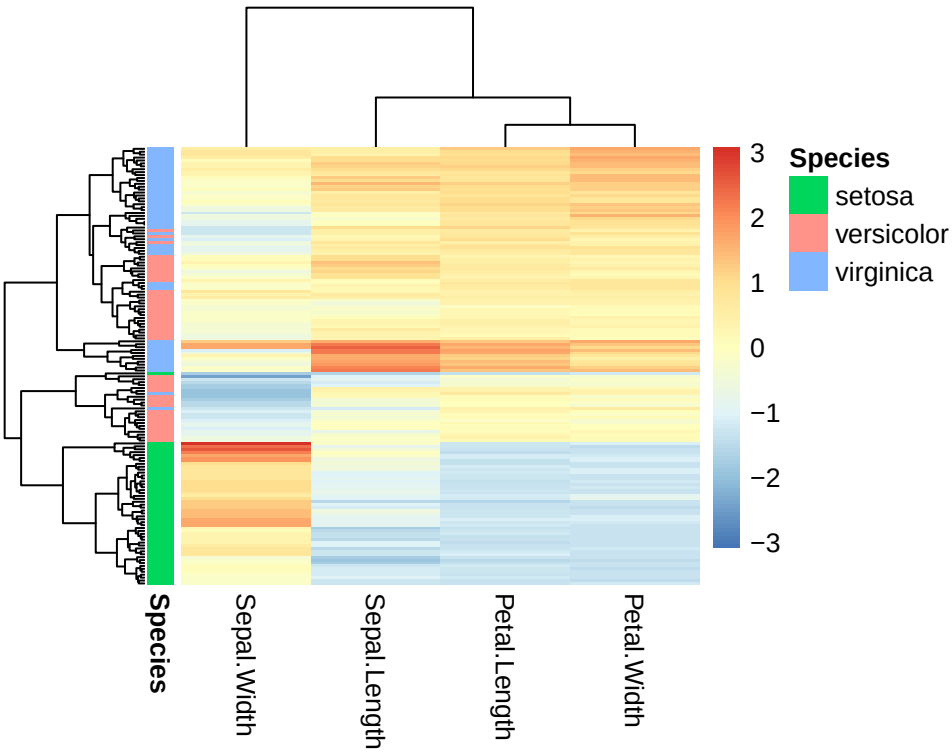
2. Now, create a pretty heatmap using `complete linkage` clustering of the rows of the data set.

```
pheatmap(plot.data, show_rownames=F, scale='column',
         clustering_method = "complete")
```



3. Annotate the rows of the heatmap with the `Species` column of the `iris` dataset. What do you observe when you compare the dendrogram and the species labels?

```r
## label the row names to be able to annotate rows
rownames(plot.data) <- 1:nrow(plot.data)

## create a data.frame for the row annotations
row.ann <- data.table(Species = iris$Species)

## plot the heatmap with complete linkage clustering
pheatmap(plot.data, annotation_row = row.ann, show_rownames=F,
         scale='column', clustering_method = "complete")
```




```r
## We observe that the complete linkage method is able to correctly classify
## the setosa species, but makes 2 subclusters for the other 2 species.
## We can explore other methods to see if they are able to distinguish between
## the different species better.
```

4. Obtain the dendogram of the row clustering using `complete linkage` clustering and partition the data into 3 clusters.

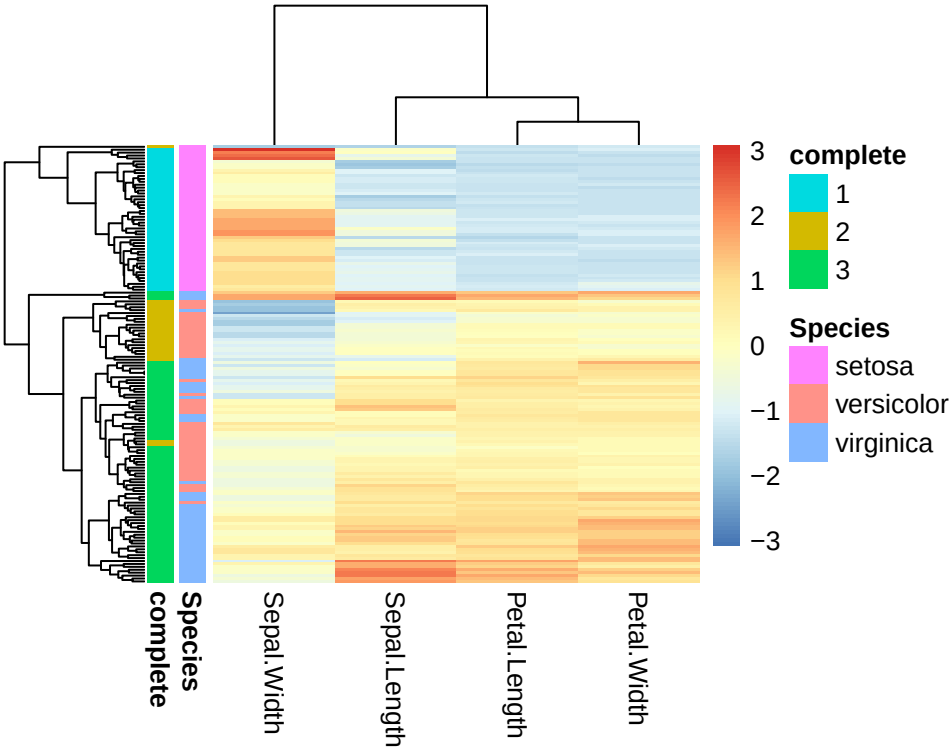
```r
## pheatmap() returns an object with dendrograms
h_complete <- pheatmap(plot.data, annotation_row=row.ann, show_rownames=F,
                scale='column', clustering_method = "complete", silent=T)
# silent=T prevent heatmap to be displayed again
complete <- cutree(h_complete$tree_row, k = 3)
complete
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   2   1   1   1   1   1   1   1   1   3   3   3   2   3   2   3   2   3   2
```

```
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   3   2   3   3   3   3   2   2   2   3   3   3   3   3   3   3   3   3   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   3   3   3   3   2   3   2   2   3   2   2   2   3   3   3   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   3   3   3   3   3   3   3   3
```

5. Create a pretty heatmap using `average` clustering of the rows annotated with the species and the complete linkage clustering results. What do you observe when you compare the dendrogram, the complete linkage results and the species labels?

```
# Add results of complete clustering to annotations
row.ann[, complete := factor(complete)]
h_average <- pheatmap(plot.data, annotation_row=row.ann, show_rownames=F,
                      scale='column', clustering_method = "average")
```




```
# We see that the average and complete linkage methods returned
# similar results (clusters). They are both able to identify the setosa
# species. And around half of the versicolor cluster with the virginica.
```

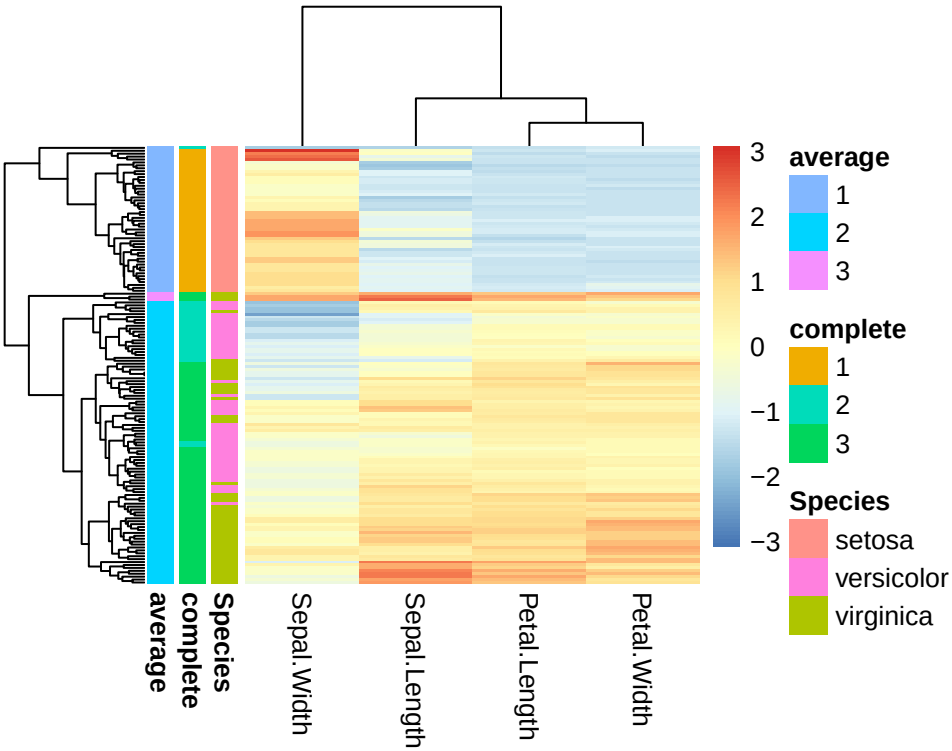
6. Partition the data into 3 clusters using the `average` clustering method.

```
average <- cutree(h_average$tree_row, k = 3)
average
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
```

```
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   2   2   2   2   2   2   2   2   3   2   2   2   2   2   2   2   3   2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   2   2   2   2   2   2   2   2   2   2   2   3   2   2   2   2   2   2   2   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   2   2   2   2   2   2   2   2
```

```
# Add results of complete and average clustering to annotations
row.ann[,average := factor(average)]
pheatmap(plot.data, annotation_row=row.ann, show_rownames=F,
                scale='column', clustering_method = "average")
```



7. Use the `table` function to compare the partitions from the complete and the average linkage clustering.

```
# The table allows us to see in how many observations the methods coincided
# for each of the 3 clusters
table(complete, average)
```

```
##          average
## complete  1  2  3
##        1 49  0  0
##        2  1 23  0
##        3  0 74  3
```

Section 03 - k-Means clustering

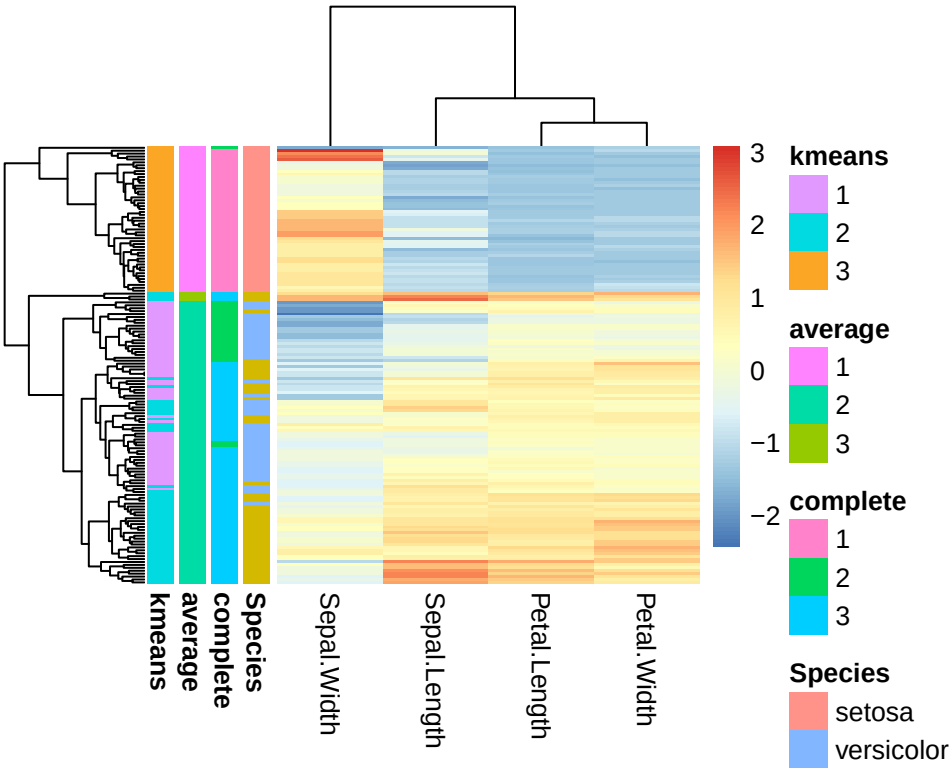
1. Perform k-means custering on the iris data set with $k = 3$.

```
scaled.data <- scale(plot.data)
km = kmeans(scaled.data, 3)
km
```

```
## K-means clustering with 3 clusters of sizes 53, 47, 50
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -0.05005221 -0.88042696    0.3465767   0.2805873
## 2   1.13217737  0.08812645    0.9928284   1.0141287
## 3  -1.01119138  0.85041372   -1.3006301  -1.2507035
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   3   3   3   3   3   3   3   3   2   2   2   1   1   1   2   1   1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   1   2   1   1   1   1   2   1   1   1   1   2   2   2   1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   1   1   1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   1   2   2   2   2   1   2   2   2   2   2   2   1   1   2   2   2   2   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   2   1   2   1   2   2   1   2   2   2   2   2   2   1   1   2   2   2   1   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   1   2   2   2   1   2   2   1
##
## Within cluster sum of squares by cluster:
## [1] 44.08754 47.45019 47.35062
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

2. Create a pretty heatmap using `average` clustering of the rows annotated with the species, both hierarchical clustering results and the k-means results. What do you observe when you compare the dendrogram, the k-means results and the species labels?

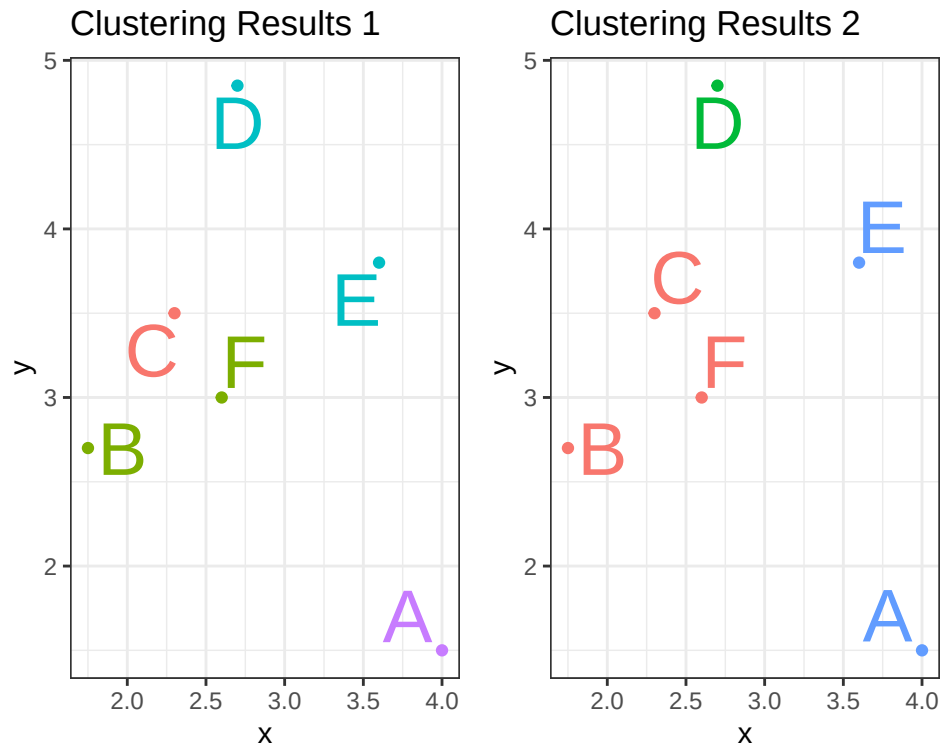
```
row.ann[, kmeans := factor(km$cluster)]
pheatmap(scaled.data, annotation_row=row.ann, show_rownames=F,
         clustering_method = "average")
```




```
## k-means is also able to distinguish setosa,
## It separates the other two species better than the
## Hierarchical clustering methods but not perfectly.
```

Section 04 - Cluster comparison

1. Compute the Rand index between the two following clustering results from two different clustering algorithms.




```
## Rand index is
## (1+10)/15 = 0.733


# Computation
# PAIR  | same cluster in both methods | different clusters in both methods
# {A,B} | 0 | 1
# {A,C} | 0 | 1
# {A,D} | 0 | 1
# {A,E} | 0 | 0
# {A,F} | 0 | 1
# {B,C} | 0 | 0
# {B,D} | 0 | 1
# {B,E} | 0 | 1
# {B,F} | 1 | 0
# {C,D} | 0 | 1
# {C,E} | 0 | 1
# {C,F} | 0 | 0
# {D,E} | 0 | 0
# {D,F} | 0 | 1
# {E,F} | 0 | 1
# ===============
# TOTAL | 1 | 10

# --> Rand index is (1+10) / (6*5/2) = 0.733
```

2. Compute the Rand indices between the clustering results from the previous sections (complete, average and k-means) and species label. *Hint:* `rand.index()` from the library `fossil`.

```
#install.packages("fossil")
library(fossil)
```

```
rand.index(complete, complete)
```

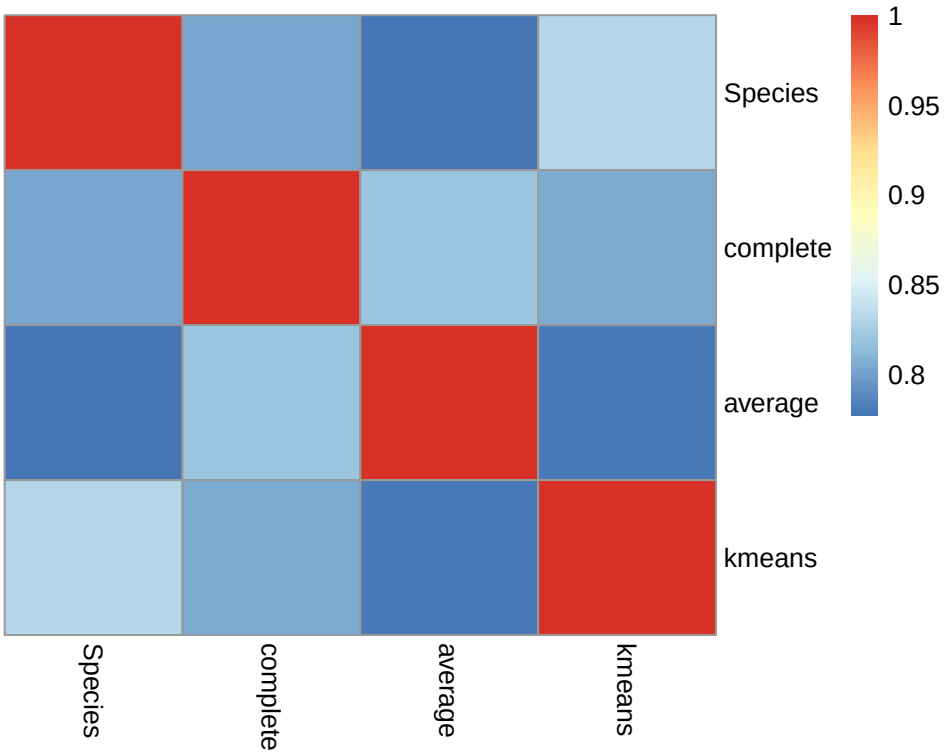
```
## [1] 1
```

```
## Convert Species to numeric
row.ann[, Species:= as.numeric(Species)]

## compute all pairwise rand indices
rand <- apply(row.ann, 2, function(i)
  apply(row.ann, 2, function(j) rand.index(as.numeric(i), as.numeric(j))))
rand
```

```
##            Species  complete   average    kmeans
## Species  1.0000000 0.8021477 0.7770917 0.8322148
## complete 0.8021477 1.0000000 0.8213870 0.8056376
## average  0.7770917 0.8213870 1.0000000 0.7795078
## kmeans   0.8322148 0.8056376 0.7795078 1.0000000
```

3. [OPTIONAL] Visualize the pair wise Rand indices with a pretty heatmap. What is the best clustering in this scenario according to the computed Rand indices?

```
pheatmap(rand, cluster_cols = F, cluster_rows = F)
```




```
rand_dt <- data.table(rand, keep.rownames = 'Clustering1') %>% melt(id.vars='Clustering1',
                                                                    value.name='rand_index',
                                                                    variable.name='Clustering2')

rand_dt[rand_index<1 & Clustering1=='Species'][which.max(rand_index)]
```

```
##    Clustering1 Clustering2 rand_index
## 1:     Species      kmeans  0.8322148
```

4. [OPTIONAL] Implement a function that computes the Rand index from two cluster label vectors (of same

length). Verify the implemented function by comparing the rand indices computed before to the rand indices that can be computed with your function.

```
my_rand_index <- function(cl1, cl2) {
  ## enumerate all pairs
  pairs = lapply(1:(length(cl1) - 1), function(i)
    lapply((i + 1):length(cl1), function(j) return(c(i, j))))
  pairs = t(matrix(unlist(pairs), nrow=2))

  ## label pairs as same or different in cl1
  same.cl1 = cl1[pairs[,1]] == cl1[pairs[,2]]

  ## label pairs as same or different in cl2
  same.cl2 = cl2[pairs[,1]] == cl2[pairs[,2]]

  ## compare the labels
  same.in.both = sum(same.cl1 & same.cl2)
  diff.in.both = sum(!same.cl1 & !same.cl2)

  ## compute the Rand index
  return((same.in.both + diff.in.both) / nrow(pairs))
}

# Compare results of one clustering to itselft for verification
my_rand_index(complete, complete)
```

```
## [1] 1
```

```
# compute rand indices and check if they are equal to the ones computed before
my_rand <- apply(row.ann, 2, function(i)
  apply(row.ann, 2, function(j) my_rand_index(as.numeric(i), as.numeric(j))))
my_rand
```

```
##            Species  complete   average    kmeans
## Species  1.0000000 0.8021477 0.7770917 0.8322148
## complete 0.8021477 1.0000000 0.8213870 0.8056376
## average  0.7770917 0.8213870 1.0000000 0.7795078
## kmeans   0.8322148 0.8056376 0.7795078 1.0000000
```

```
tol <- 1e-5
all(abs(rand-my_rand) <= tol)
```

```
## [1] TRUE
```

Section 05 - Dimensionality reduction with PCA

1. Let X be the `iris` data set without the `Species` column and only for the species setosa. Perform PCA on X. Make sure that you scale and center the data before performing PCA.

```
## get the data
data(iris)
iris_dt <- as.data.table(iris)
pca_data <- iris_dt[Species == "setosa", -"Species"]
pca <- prcomp(pca_data, center=TRUE, scale.=TRUE)
pca
```

```
## Standard deviations (1, .., p=4):
```

```
## [1] 1.4347614 1.0110283 0.8172027 0.5014592
##
## Rotation (n x k) = (4 x 4):
##                     PC1        PC2           PC3         PC4
## Sepal.Length -0.6044164  0.3349908 -0.0673598261 -0.71966982
## Sepal.Width  -0.5756194  0.4408461 -0.0007138239  0.68870645
## Petal.Length -0.3754348 -0.6269717 -0.6770628102  0.08683986
## Petal.Width  -0.4029788 -0.5480350  0.7328356536  0.01475204
```

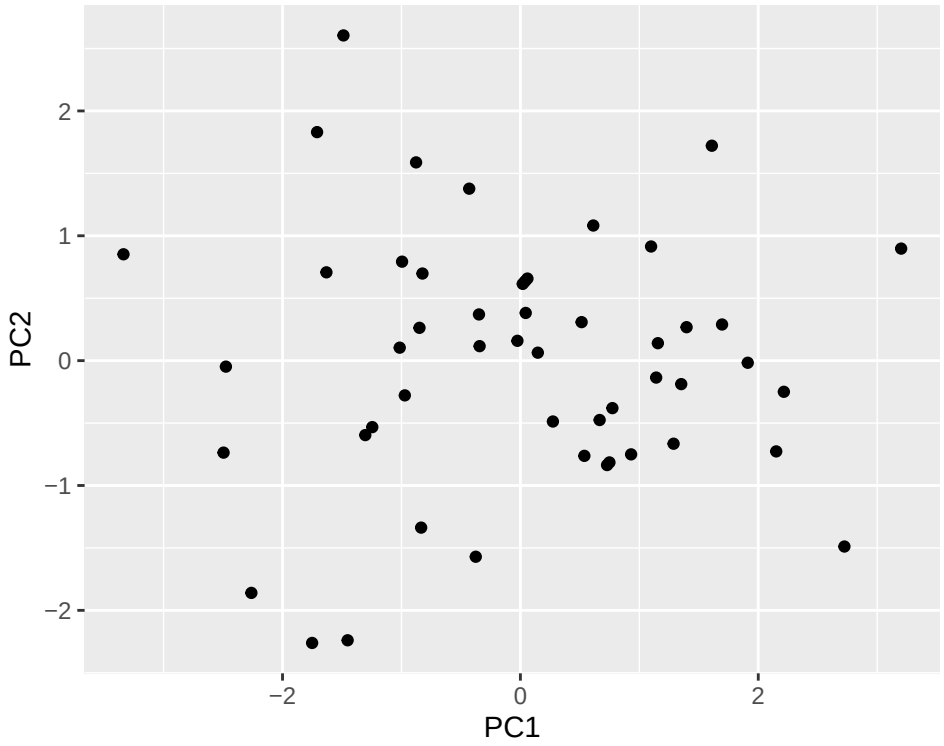
2. Which proportion of the variance is explained by each principle component?

```
summary(pca)
```

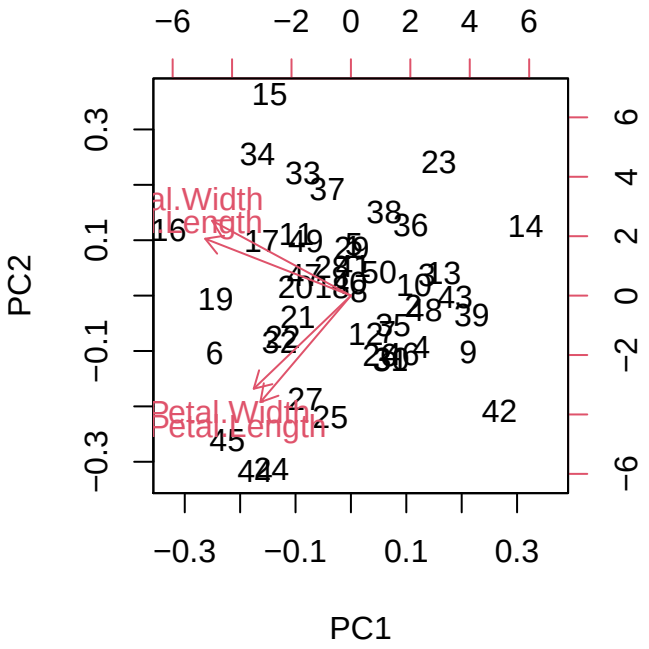
```
## Importance of components:
##                           PC1    PC2    PC3     PC4
## Standard deviation     1.4348 1.0110 0.8172 0.50146
## Proportion of Variance 0.5146 0.2555 0.1670 0.06287
## Cumulative Proportion  0.5146 0.7702 0.9371 1.00000
```

3. Compute the projection of X from the PCA result and plot the projection on the first two principle components. *Hint:* `predict()` Additionally look at the biplot and come up with an interpretation of the first principal component.

```
# Compute projection of the data using the predict function.
proj <- as.data.table(predict(pca))
ggplot(proj, aes(PC1, PC2)) + geom_point()
```



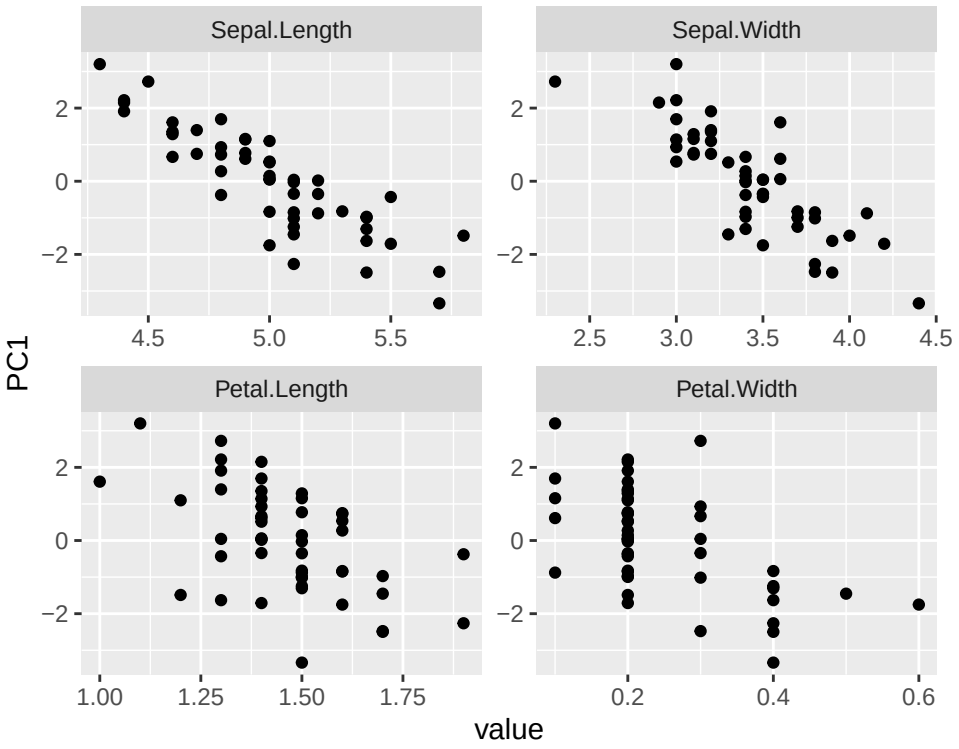

```
# The biplot can be shown the following way:
biplot(pca)
```




```
# In the biplot one can observe, that all lengths and widths contribute with the same sign to the PC1.
# PC1 correlates negatively with the projection of the axes of the variables
# in the original dataset (red vectors)

# An increase in the size of the flower implies a decrease in the projected PC1 value.

# We can assume that the PC1 describes the size of the flower.
```

4. Plot the first principal component against the other variables in the dataset and discuss whether this supports your previously stated interpretation. Discuss the interpretation in your Breakout Room.

```
pc_iris <- cbind(iris_dt[Species == "setosa"], proj)
pc_iris <- melt(pc_iris, id.vars = c("Species", 'PC1', 'PC2', 'PC3', 'PC4'))

ggplot(pc_iris, aes(value, PC1)) + geom_point() + facet_wrap(~variable, scales = 'free')
```

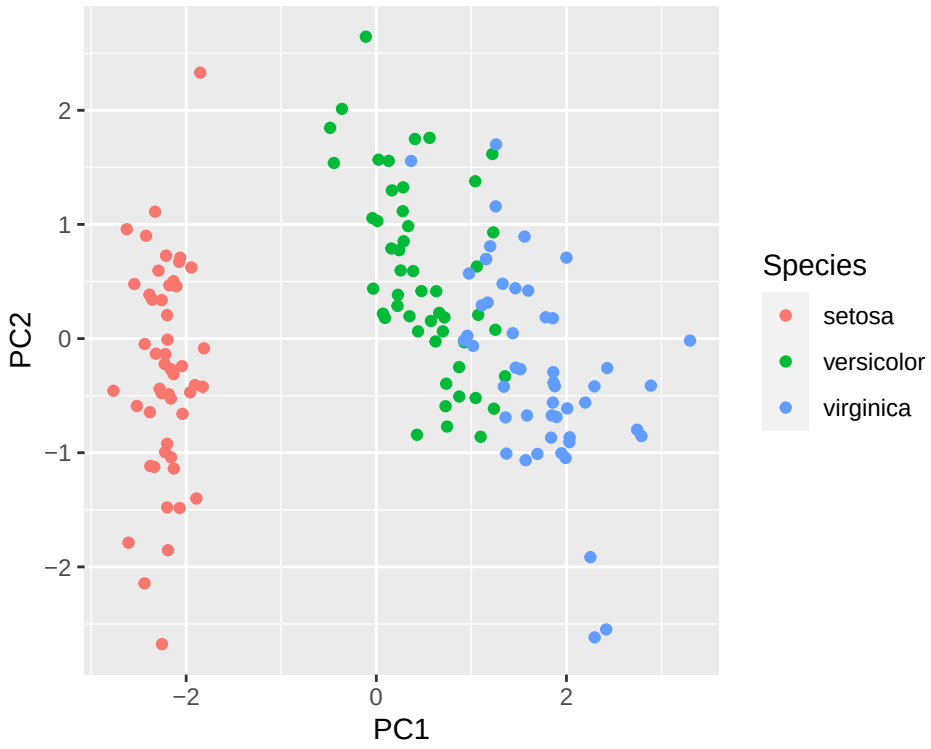


5. Repeat the steps 1 - 4 for all species jointly (not only setosa). Discuss whether your original interpretation of the first principal component changed when performing the PCA for all species jointly. Use color to differentiate between the species in your plots.

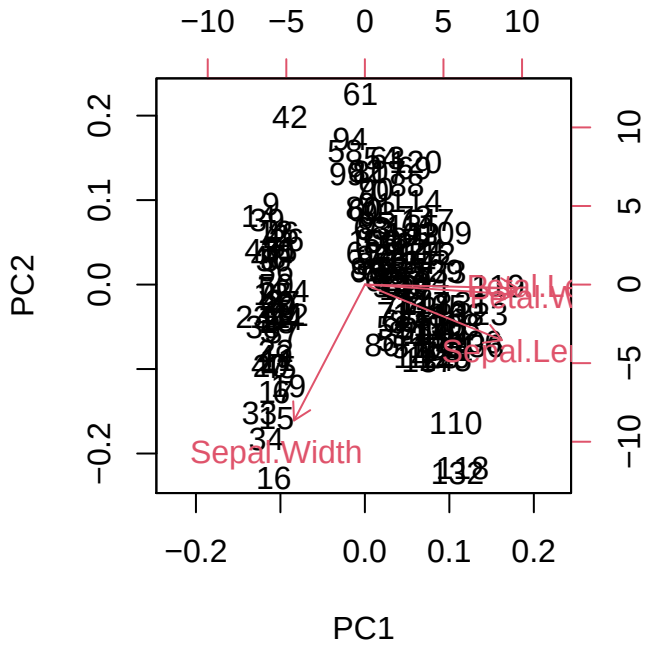
```
# First run the pca on all species
pca_data <- iris_dt[, -"Species"]
pca <- prcomp(pca_data, center=TRUE, scale.=TRUE)
pca
```

```
## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
#
# Compute projection of the data using the predict function.
proj <- as.data.table(predict(pca))
pc_iris <- cbind(iris_dt, proj)
ggplot(pc_iris, aes(PC1, PC2, color = Species)) + geom_point()
```

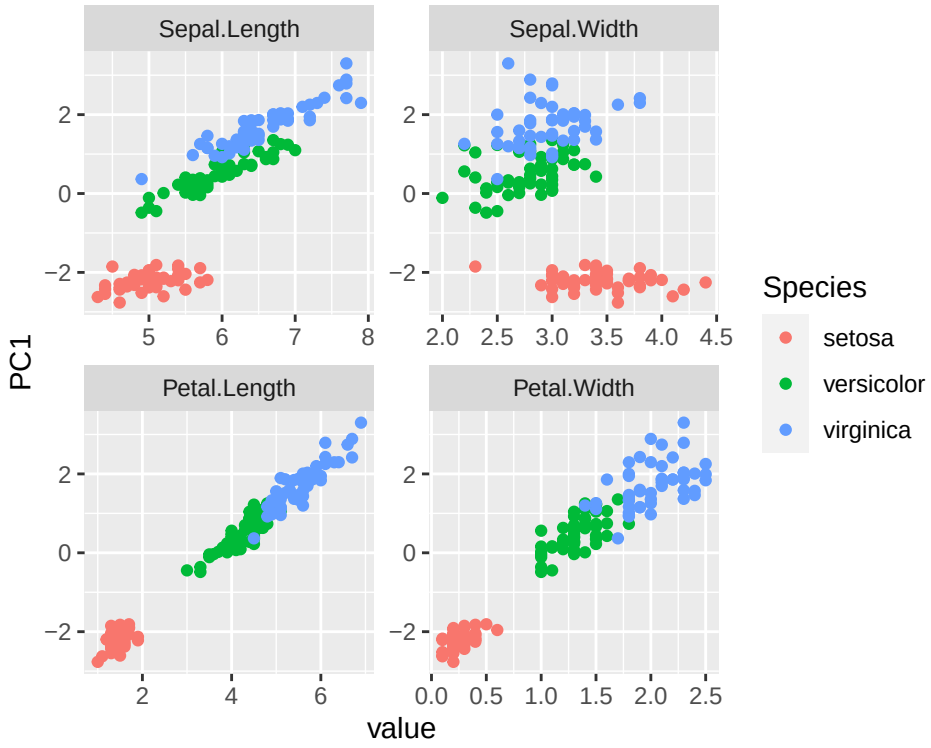



```
# In the plot above one can see that the first principal component clearly seperates the species.

# The biplot can be shown the following way:
biplot(pca)
```




```
# In the biplot one can observe that the lengths and widths do not contribute with the
# same sign any longer.

pc_iris <- melt(pc_iris, id.vars = c("Species", 'PC1', 'PC2', 'PC3', 'PC4'))
ggplot(pc_iris, aes(value, PC1, color=Species)) +
```

```
    geom_point() +
    facet_wrap(~variable, scales = 'free')
```




```
# The PC1 strongly associates with the size of the petal wich strongly associates with the species.
# Additionally, it is negatively correlated with the sepal width,
# as the sepal width is higher for setosa than for the other species.
# Therefore, the PC1 can be interpreted as describing the species of each flower.
```